Appendix G: Test & Data Message source

This section lists snippets of the source code (developed at the University of Washington) used to construct Test Messages 1, 2 & 3, and Data Messages 1, 2 & 3.

Test Message 1 snippet:

```
/* write the message block id to the argos stream */
ArgosPutByte(ArgosBlkId);

/* write the month to the argos data stream */
byte = (FwRev>>16)&0xff; ArgosPutByte(byte);

/* write the day to the argos data stream */
byte = (FwRev>>8)&0xff;  ArgosPutByte(byte);

/* write the year to the argos data stream */
byte = FwRev&0xff;       ArgosPutByte(byte);

/* write the float id to the argos stream */
ArgosPutWord(mission.FloatId);

/* write the interval timer to the argos data stream */
u=(unsigned int)itimer(); ArgosPutWord(u);

/* get the current pressure */
if (GetP(&f)<=0)  {f=NaN();}

/* write the status bits to the argos data stream */
ArgosPutWord(vitals.status);

/* write the current pressure to the argos stream */
ArgosPutWord(EncodeP(f));

/* write the vacuum reading to the argos data stream */
ArgosPutByte(vitals.Vacuum);

/* write the current air-bladder pressure to the argos data stream */
ArgosPutByte(BarometerAd8());

/* write the open-circuit voltage */
ArgosPutByte(BatVoltsAd8());
```

```
/* write the up-time to the argos data stream */
byte=mission.TimeUp/TQuantum; ArgosPutByte(byte);

/* write the down-time to the argos data stream */
u=mission.TimeDown/TQuantum; ArgosPutWord(u);

/* write the park pressure to the argos data stream */
ArgosPutWord(EncodeP(mission.PressurePark));

/* write the piston park position to the argos data stream */
ArgosPutByte(mission.PistonParkPosition);

/* write the buoyancy nudge to the argos data stream */
ArgosPutByte(mission.PistonBuoyancyNudge);

/* write the OK vacuum count to the argos data stream */
ArgosPutByte(mission.OkVacuumCount);

/* write the ascent time-out to the argos data stream */
byte=mission.TimeOutAscent/TQuantum; ArgosPutByte(byte);

/* write the target bladder pressure to the argos data stream */
ArgosPutByte(mission.MaxAirBladder);

/* write the profile pressure to the argos data stream */
ArgosPutWord(EncodeP(mission.PressureProfile));

/* write the deep-profile piston position to the argos data stream */
ArgosPutByte(mission.PistonDeepProfilePosition);

/* write the PnP cycle length to the argos data stream */
ArgosPutByte(mission.PnpCycleLength);

/* complete message #1 with filler bytes */
while (ArgosFifoLen()<MsgLen) {ArgosPutByte(0xff);}
```

Test Message 2 snippet:

```
/* Msg2: write the message block id to the argos stream */
ArgosPutByte(ArgosBlkId);

/* Msg2: write the month to the argos data stream */
byte = (FwRev>>16)&0xff; ArgosPutByte(byte);

/* Msg2: write the day to the argos data stream */
```

```
byte = (FwRev>>8)&0xff;  ArgosPutByte(byte);

/* Msg2: write the year to the argos data stream */
byte = FwRev&0xff;       ArgosPutByte(byte);

/* Msg2: write piston full-extension position to the argos data stream */
ArgosPutByte(mission.PistonFullExtension);

/* Msg2: write piston full-retraction position to the argos data stream */
ArgosPutByte(mission.PistonFullRetraction);

/* Msg2: write initial piston buoyancy nudge to the argos data stream */
ArgosPutByte(mission.PistonInitialBuoyancyNudge);

/* Msg2: write park-level piston hyper-retraction for N2 floats */
ArgosPutByte(mission.PistonParkHyperRetraction);

/* Msg2: write the piston position for pressure activation mode */
ArgosPutByte(mission.PistonPActivatePosition);

/* Msg2: write deep-profile descent period to the argos data stream */
ArgosPutByte(mission.TimeDeepProfileDescent/Hour);

/* Msg2: write park descent period to the argos data stream */
ArgosPutByte(mission.TimeParkDescent/Hour);

/* Msg2: write mission prelude period to the argos data stream */
ArgosPutByte(mission.TimePrelude/Hour);

/* Msg2: write argos rep-rate to the data stream */
ArgosPutByte(mission.ArgosRepPeriod);

/* Msg2: write the argos id to the data stream */
ArgosPutByte((mission.argosid.root>>12)&0xff);
ArgosPutByte((mission.argosid.root>>4)&0xff);
ArgosPutByte((mission.argosid.root&0xf)<<4 | (mission.argosid.ext>>4)&0xf);

/* Msg2: write argos frequency as kHz off-center of 401.65MHz */
ArgosPutByte(EncodeKHz(mission.ArgosMegaHertz));

/* Msg2: write the Sbe41 serial number */
ArgosPutWord(Sbe41Serno);

/* Msg2: write the Sbe41 firmware revision */
ArgosPutWord(Sbe41Fwrev);
```

```
/* Msg 2: write the current UNIX epoch of the Apf9a RTC (little endian order) */
t=time(NULL); ArgosPut((const unsigned char *)(&t),sizeof(t));

/* Msg 2: write the ToD specification (minutes) */
if (inRange(0,mission.ToD,Day)) ArgosPutWord(mission.ToD/Min);

/* Msg 2: write a sentinel value (0xfffe) if ToD feature disabled */
else ArgosPutWord(0xfffeU);

/* Msg 2: write the debug-level */
ArgosPutWord(debugbits);
```

Data Message 1 snippet:

```
/* write the message block id to the argos stream */
ArgosPutByte(ArgosBlkId);

/* write the float id to the argos stream */
ArgosPutWord(mission.FloatId);

/* write the profile id to the argos stream */
ArgosPutByte((unsigned char)PrfIdGet());

/* write the number of observations to the argos stream */
ArgosPutByte(ObsIndex);

/* write the status word to the argos stream */
ArgosPutWord(vitals.status);

/* write the surface pressure to the argos data stream */
ArgosPutWord(EncodeP(vitals.SurfacePressure));

/* write the current pressure to the argos data stream */
ArgosPutByte(EncodePs(p));

/* write the surface piston position */
ArgosPutByte(vitals.SurfacePistonPosition);

/* write the park piston position */
ArgosPutByte(mission.PistonParkPosition);
```

```
/* write the profile piston position */
ArgosPutByte(mission.PistonDeepProfilePosition);

/* write the SBE41 status long-word to the argos stream */
ArgosPutLongWord(vitals.Sbe41Status);

/* write the quiescent volts and amps at the park level */
ArgosPutByte(vitals.QuiescentVolts); ArgosPutByte(vitals.QuiescentAmps);

/* write the SBE41 volts and amps at the park level */
ArgosPutByte(vitals.Sbe41Volts); ArgosPutByte(vitals.Sbe41Amps);

/* write the buoyancy pump volts and amps at the end of the initial pump extension */
ArgosPutByte(vitals.BuoyancyPumpVolts); ArgosPutByte(vitals.BuoyancyPumpAmps);

/* write the air-pump volts and amps */
ArgosPutByte(vitals.AirPumpVolts); ArgosPutByte(vitals.AirPumpAmps);

/* write the air bladder pressure */
ArgosPutByte(vitals.AirBladderPressure);

/* write the number of air pump pulses so far */
ArgosPutByte(vitals.AirPumpPulses);

/* write the air-pump energy paramters */
ArgosPutWord(vitals.AirPumpVoltSec);

/* complete message #1 with filler bytes */
while (ArgosFifoLen()<MsgLen) {ArgosPutByte(0xff);}
```

Data Message 2 snippet:

```
/* Msg 2: write the UNIX epoch when the down-time ended */
t=vitals.TimeDownEpoch; ArgosPut((const unsigned char *)(&t),sizeof(t));

/* Msg 2: write elapsed time to initiation of telemetry phase */
t=(time_t)(difftime(vitals.TelemetryInitEpoch,vitals.TimeDownEpoch)/Min);
ArgosPutWord((unsigned int)((t>=0)?t:(0x10000L+t)));

/* Msg 2: write the number of active-ballast adjustments */
ArgosPutByte(vitals.ActiveBallastAdjustments);
```

```
  /* Msg 2: write the park-level statistics */
  ArgosPutWord(ParkPt.n);
  ArgosPutWord(EncodeT(ParkPt.mean.t));
  ArgosPutWord(EncodeP(ParkPt.mean.p));
  ArgosPutWord(EncodeT(ParkPt.stddev.t));
  ArgosPutWord(EncodeP(ParkPt.stddev.p));
  ArgosPutWord(EncodeT(ParkPt.Tmin.t));
  ArgosPutWord(EncodeP(ParkPt.Tmin.p));
  ArgosPutWord(EncodeT(ParkPt.Tmax.t));
  ArgosPutWord(EncodeP(ParkPt.Tmax.p));
  ArgosPutWord(EncodeP(ParkPt.Pmin));
  ArgosPutWord(EncodeP(ParkPt.Pmax));

  /* Msg 2: complete message #2 with filler bytes */
  while (ArgosFifoLen()<2*MsgLen) {ArgosPutByte(0xff);}
```

Data Message 3 snippet:

```
  /* Msg 3: write the vacuum reading at the park level */
  ArgosPutByte(vitals.Vacuum);

  /* Msg 3: conditon the pump-time for an unsigned int */
  if (vitals.BuoyancyPumpOnTime<0) vitals.BuoyancyPumpOnTime=0;
  else if (vitals.BuoyancyPumpOnTime>UINT_MAX)
vitals.BuoyancyPumpOnTime=UINT_MAX;

  /* Msg 3: write the pump motor time */
  ArgosPutWord((unsigned int)vitals.BuoyancyPumpOnTime);

  /* Msg 3: write the hydro-sample at the park level */
  ArgosPutWord(EncodeT(vitals.ParkObs.t));
  ArgosPutWord(EncodeS(vitals.ParkObs.s));
  ArgosPutWord(EncodeP(vitals.ParkObs.p));

  /* write the hydrographic profile to the argos stream */
  for (i=0; i<ObsIndex && i<pTableSize; i++)
  {
    ArgosPutWord(EncodeT(obs[i].t));
    ArgosPutWord(EncodeS(obs[i].s));
    ArgosPutWord(EncodeP(obs[i].p));
  }
```

```
/* check to see if there's room to write the auxiliary engineering data */
if ((ArgosFifoLen()%MsgLen) && (N=(MsgLen-(ArgosFifoLen()%MsgLen)))>=2)
{
  /* write the pressure divergence as auxiliary engineering data */
  ArgosPutWord(EncodeP(vitals.PDivPts)); N-=2;

  /* check to see if there's room to write more auxiliary engineering data */
  if (N>=2)
  {
    /* write elapsed time to initiation of telemetry phase */
    t=(time_t)(difftime(vitals.ProfileInitEpoch,vitals.TimeDownEpoch)/Min);
    ArgosPutWord((unsigned int)((t>=0)?t:(0x10000L+t))); N-=2;

    /* check room to write the number of descent marks */
    if (N>=1) {ArgosPutByte(vitals.ParkDescentPCnt); N--;}

    /* loop through each descent mark */
    for (i=0; i<N && i<vitals.ParkDescentPCnt && i<ParkDescentPMax; i++)
    {
      /* write the current descent mark (bars) to the argos fifo */
      ArgosPutByte(vitals.ParkDescentP[i]);
    }
  }
}
```